k-NN Boosting Prototype Learning for Object Classification

Paolo Piro, Michel Barlaud, Richard Nock, Frank Nielsen

K-NN BOOSTING PROTOTYPE LEARNING FOR OBJECT CLASSIFICATION

Paolo Piro[1], Michel Barlaud[1], Richard Noch[2], Frank Nielsen[3]

[1]CNRS / University of Nice-Sophia Antipolis, France
[2]CEREGMIA, University of Antilles-Guyane, France
[3]LIX, Ecole Polytechnique, France

ABSTRACT

Object classification is a challenging task in computer vision. Many approaches have been proposed to extract meaningful descriptors from images and classifying them in a supervised learning framework. In this paper, we revisit the classic k-nearest neighbors (k-NN) classification rule, which has shown to be very effective when dealing with local image descriptors. However, k-NN still features some major drawbacks, mainly due to the uniform voting among the nearest prototypes in the feature space. In this paper, we propose a generalization of the classic k-NN rule in a supervised learning (boosting) framework. Namely, we redefine the voting rule as a strong classifier that linearly combines predictions from the k closest prototypes. To induce this classifier, we propose a novel learning algorithm, MLNN (Multiclass Leveraged Nearest Neighbors), which gives a simple procedure for performing prototype selection very efficiently. We tested our method on 12 categories of objects, and observed significant improvement over classic k-NN in terms of classification performances.

1. INTRODUCTION

In this paper, we address the task of multiclass object categorization. It consists in automatically classifying an unlabeled region extracted from an image (e.g., by segmentation) according to a set of predefined objects. This task is very challenging, and is attracting more and more research effort from the computer vision community, as prompted by the plethora of classification approaches proposed for PASCAL 2009 competition[1]. A wide range of image descriptors has been investigated for object categorization purposes, which generally rely on detecting relevant local characteristics of objects (e.g., local shape and appearance). The best known examples of such descriptors are SIFT [1], which are typically combined into Bags-of-Features [2] or Pyramid [3] representations.

Despite lots of works, much remains to be done to challenge human level performances. In fact, images carry only parts of the information that is used by humans to recognize objects, and parts of the information available from images may be highly misleading: for example, real object categories may exhibit high intra-class variability (*i.e.*, visually different objects may belong to the same category) and low inter-class variability (*i.e.*, distinct categories may contain visually similar objects).

Voting classification techniques, like k-nearest neighbors (k-NN), have been shown to be very effective when dealing with local image descriptors. However, they may suffer from high sensitivity to "noisy" prototypes, thus requiring suitable learning procedures for rejecting unreliable matches. Moreover, it is a critical challenge to reduce the computational cost of descriptor matching without impairing classification performances. In order to cope with these issues, the literature has favored two main approaches so far: improve categorization by means of local classifiers [4, 5, 6], or filter out ill-defined examples [7].

In this paper, we propose a novel solution: a new provable boosting algorithm for nearest-neighbors (NN) rules in a multiclass framework. Our algorithm, MLNN (Multiclass Leveraged Nearest Neighbors), induces a multiclass leveraged nearest neighbors rule that generalizes the uniform k-NN rule, using directly the examples as weak hypotheses. Finally, the most significant advantage of MLNN lies in its ability to find out the most relevant prototypes for categorization, thus enabling to filter out the remaining examples.

In the following section we present MLNN, along with the statement of its theoretical properties. Then, we present and discuss experimental results of object categorization.

2. METHOD

2.1. Problem statement and notations

Instead of splitting the multiclass classification problem in as many *one-versus-all* (two-class) problems — a frequent approach in boosting [8] — we directly tackle the *multiclass* problem, following [9]. For a given query, we compute its *classification score* for all categories (or classes, or labels). Then, we select the label with the maximum score. We suppose given a set $\mathcal{S}$ of m annotated descriptors arising from images (or image regions). Each image descriptor provides a

[1]http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2009/

training *example* $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is the image feature vector and $\boldsymbol{y}$ the *class vector* that specifies the category membership of the descriptor. In particular, the sign of component y_c gives the positive/negative membership of the example to class c ($c = 1, 2, ..., C$). Inspired by the multiclass boosting analysis of [9], we constrain the class vector to be *symmetric*, *i.e.*, $\sum_{c=1}^{C} y_c = 0$ by setting: $y_{\tilde{c}} = 1$, $y_{c \neq \tilde{c}} = -\frac{1}{C-1}$, where $\tilde{c}$ is the true image category.

2.2. (Leveraged) Nearest Neighbors

The regular k-NN rule is based on majority vote among the k nearest neighbors in set $\mathcal{S}$, to decide the class of query $\boldsymbol{x}$. It can be defined as the following multiclass classifier $\boldsymbol{h} = \{h_c,\ c = 1, 2, ..., C\}$:

$$h_c(\boldsymbol{x}) = \frac{1}{k} \sum_{i \sim_k x} [y_{ic} > 0] \ , \tag{1}$$

where $h_c \in [0, 1]$ is the classification score for class c, $i \sim_k x$ denotes an example $(\boldsymbol{x_i}, \boldsymbol{y_i})$ belonging to the k nearest neighbors of $\boldsymbol{x}$ and square brackets denote the indicator function.

In this paper, we propose to generalize (1) to the following *leveraged* k-NN rule $\boldsymbol{h}^\ell = \{h_c^\ell,\ c = 1, 2, ..., C\}$:

$$h_c^\ell(\boldsymbol{x}) = \sum_{j \sim_k x} \alpha_j y_{jc} \in \mathbb{R} \ , \tag{2}$$

where the k nearest neighbors are searched either in $\mathcal{S}$, or in a sparse subset $\mathcal{P} \subseteq \mathcal{S}$ obtained after a *prototype selection* step, achieved before any query is presented. Each example in $\mathcal{P}$ is a relevant category *prototype*. Prototype selection is achieved by using the leveraging coefficients α_j (2), which are expected to represent their "confidence " for classifying new data.

In the following sections we describe the boosting-like procedure we propose to compute the α_j's. In particular, we propose to minimize a particular upperbound of the risk functional on training data, thus exploiting a very important trick that has been at the center of major advances in classification over the last ten years.

2.3. Multiclass surrogate risk minimization

In order to fit our classification rule (2) onto training set $\mathcal{S}$, we focus on the minimization of a multiclass exponential (surrogate[2]) risk:

$$\varepsilon^{\exp}\left(\boldsymbol{h}^\ell, \mathcal{S}\right) \doteq \frac{1}{m} \sum_{i=1}^{m} \exp \quad -\frac{1}{C} \sum_{c=1}^{C} y_{ic} h_c^\ell(\boldsymbol{x_i}) \Bigg) \ . \tag{3}$$

This function is an upper bound of the *empirical risk*:

$$\varepsilon^{0/1}\left(\boldsymbol{h}^\ell, \mathcal{S}\right) \doteq \frac{1}{mC} \sum_{i=1}^{m} \sum_{c=1}^{C} \left[y_{ic} h_c^\ell(\boldsymbol{x_i}) < 0\right] \ , \tag{4}$$

[2]We call *surrogate* a function that upperbounds the risk functional we should minimize, and thus can be used as a primer for its minimization.

which is not differentiable and often computationally hard to directly minimize [10]. Remark that both risks (3, 4) depend on quantity $y_{ic} h_c^\ell(\boldsymbol{x_i})$, the *edge* of classifier $\boldsymbol{h}^\ell$ on example $(\boldsymbol{x_i}, \boldsymbol{y_i})$ for class c. This edge is positive iff the category membership predicted by the classifier agrees with the true membership of the example. Plugging definition (2) into surrogate risk (3) gives:

$$\varepsilon^{\exp}\left(\boldsymbol{h}^\ell, \mathcal{S}\right) \doteq \frac{1}{m} \sum_{i=1}^{m} \exp\left(-\sum_{j=1}^{m} \alpha_j \mathrm{r}_{ij}\right) \ , \tag{5}$$

which highlights an essential ingredient of our algorithm, *i.e.* the *multiclass* k-NN *edge matrix* $[\mathrm{r}_{ij}]_{m \times m}$, whose entry r_{ij} is different from zero iff example j is a neighbor of i, whereas the positive (negative) sign of r_{ij} specifies the membership of the two examples to the same (not the same) class. (See definition (7) in Alg. 1.) Finally, after computing the edge matrix, which is a constant term in (5), the unknown leveraging coefficients α_j can be fitted by running the algorithm described in the following section, which iteratively minimizes the surrogate risk.

2.4. MLNN: Multiclass Leveraged k-NN rule

Algorithm Pseudocode of MLNN is shown in Alg. 1. Like common boosting algorithms, MLNN operates on a set of weights w_i ($i = 1, 2, ..., m$) defined over training data. These weights are repeatedly updated based on δ_j, which is a local measure of the class density around a given example j. Namely, at each iteration t of the algorithm, a *weak index chooser* oracle $\textsc{Wic}(\{1, 2, ..., m\}, t)$ determines index $j \in \{1, 2, ..., m\}$ of the example to leverage (step I.0). Various choices are possible for this oracle. The simplest is perhaps to compute Eq. (10, 11) for all the training examples, then to pick j maximizing δ_j:

$$j \quad \textsc{Wic}(\{1, 2, ..., m\}, t) \ : \ \delta_j = \max_{j \in \{1, 2, ..., m\}} \delta_j^t \ . \tag{6}$$

Furthermore, notice that, when whichever w_j^+ or w_j^- is zero, δ_j in (11) is not finite. We propose a simple strategy to eliminate this drawback, inspired by [8], *i.e.*, to add $1/m$ to both the numerator and the denominator of the fraction in the log term of (11). This smoothes out δ_j, guaranteeing its finiteness without impairing convergence of MLNN.

Complexity MLNN shares the property with boosting algorithms of being resources-friendly: since computing the leveraging coefficients scales linearly with the number of neighbors, its time complexity bottleneck does not rely on boosting, but on the complexity of nearest neighbor search. Furthermore, its space complexity is also reduced: since weak hypotheses are examples, example j can be a classifier only for its *reciprocal* nearest neighbors — those examples for which j itself is a neighbor —, corresponding to non-zero entries in column j of edge matrix (7). This matrix is thus extremely

Algorithm 1: MULTICLASS LEVERAGED k-NN MLNN($\mathcal{S}$)

Input: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\,, \quad i = 1, 2, ..., m\,, \quad \boldsymbol{y_i} \in \{-\frac{1}{C-1}, 1\}^C\}$

Let $\mathrm{r}_{ij} \doteq \begin{cases} \frac{1}{C}\sum_{c=1}^{C} y_{ic}y_{jc} & \text{if} \quad j \sim_k i \\ 0 & \text{otherwise} \end{cases}$ (9)

Let $\alpha_j \quad 0, \quad \forall j = 1, 2, ..., m$

Let $w_i \quad 1/m, \quad \forall i = 1, 2, ..., m$

for $t = 1, 2, ..., T$ **do**

[I.0] Weak index chooser oracle:

Let $j \quad \textsc{Wic}(\{1, 2, ..., m\}, t)$;

[I.1] Let

$$w_j^+ = \sum_{i:\ \mathrm{r}_{ij}>0} w_i, \quad w_j^- = \sum_{i:\ \mathrm{r}_{ij}<0} w_i\,, \qquad (10)$$

$$\delta_j \quad \frac{(C-1)^2}{C} \log \quad \frac{(C-1)w_j^+}{w_j^-}\Bigg)\,; \qquad (11)$$

[I.2] Let

$$w_i \quad w_i \exp(-\delta_j \mathrm{r}_{ij}), \quad \forall i\ :\ j \sim_k i\,; \qquad (12)$$

[I.3] Let $\alpha_j \quad \alpha_j + \delta_j$

Output: $h_c^\ell(\boldsymbol{x}) = \sum_{j \sim_k x} \alpha_j y_{jc}, \quad \forall c = 1, 2, ..., C$

sparse for reasonable values of k. As a consequence, update rule (12) is to be computed on a small number of examples.

Convergence Using known arguments of the boosting theory [10], we proved the convergence of MLNN to the minimum of the surrogate risk, along with a convergence rate, which is based on the following *weak index assumption* (**WIA**):

WIA: let $p_j \doteq w_j^+/(w_j^+ + w_j^-)$. There exist some $\gamma > 0$ and $\eta > 0$ such that the following two inequality holds for index j returned by $\textsc{Wic}(\{1, 2, ..., m\}, t)$:

$$|p_j - 1/C| \quad \geq \quad \gamma\,, \qquad (7)$$

$$(w_j^+ + w_j^-)/||\boldsymbol{w}||_1 \quad \geq \quad \eta\,. \qquad (8)$$

We summarize this fundamental convergence property in the following theorem:

Theorem 1 *If the **WIA** holds for $\tau \leq T$ steps, then* MLNN *converges with τ to $\boldsymbol{h}^\ell$ realizing the **global** minimum of the surrogate risk (3), and $\varepsilon^{0/1}(\boldsymbol{h}^\ell, \mathcal{S}) \leq \exp(-\frac{C}{C-1}\eta\gamma^2\tau)$.*

Inequality (7) is the usual weak learning assumption, used to analyze classical boosting algorithms [11, 8], when considering examples as weak classifiers. A *weak coverage assumption* (8) is needed as well, because insufficient *coverage* of the reciprocal neighbors could easily wipe out the surrogate risk reduction due to a large γ in (7). For a deeper insight into the properties of our k-NN boosting method, see [12].

Fig. 1. Twelve categories from the Caltech-101 database.

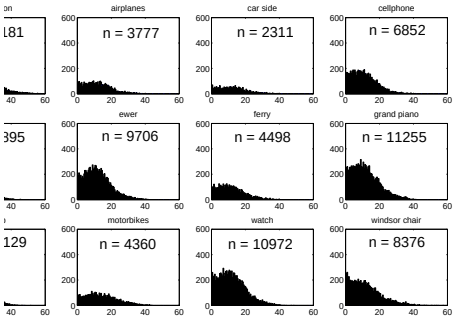

Fig. 2. Histograms of prototype leveraging coefficients α_j per category. The overall number of prototypes in each category is reported as well.

3. EXPERIMENTS

In this section we present experimental results of MLNN vs plain k-NN on a database of real objects. We carried out experiments in order to investigate the improvements brought by boosting on nearest neighbor voting. Namely, we used 12 categories from the well-known Caltech-101 database for object classification: *accordion, airplanes, car side, cellphone, cup, ewer, ferry, grand piano, laptop, motorbikes, watch, Windsor chair* (Fig. 1). This database contains a large variety of objects, and also exhibits high intra-class variability, i.e., visually different objects may be in the same category.

3.1. Training

We used 40 training images per category, and extracted dense SIFT descriptors [1] from image regions corresponding to objects. For this purpose, we used the ground-truth object masks provided with the database. We computed dense descriptors of 16×16 patches over a grid with spacing of 8 pixels, as proposed in [3]. We used the descriptors of all training objects for learning prototypes, i.e., a subset of relevant object descriptors with their leveraging coefficients. Namely, we retained only examples with positive α_j as prototypes for classifying test images. In Fig. 2 we show how values of prototype leveraging coefficients are distributed in each category. The best represented categories are those maximizing the integral of such histograms, i.e., those containing most of the prototypes with the largest coefficient values. We used all non-training images (2,039 overall) to test classification performances of MLNN.

3.2. Classification

In order to obtain a global classification score for a given test image $\boldsymbol{X}$, each of the query descriptors $\boldsymbol{x} \in \boldsymbol{X}$ was first classified independently by our leveraged k-NN rule (2). Because prototype classes are highly imbalanced, as displayed in Fig. 2, we smoothed out aggregate scores with a standard technique [13]. Hence, we predict label $\hat{c}$ for a query image $\boldsymbol{Q}$ as follows:

$$\hat{c}(\boldsymbol{X}) \doteq \arg\max_{c} \frac{1}{m_c^{\mathcal{P}}} \sum_{\boldsymbol{x} \in \boldsymbol{X}} h_c^{\ell}(\boldsymbol{x}) \,, \qquad (13)$$

where $m_c^{\mathcal{P}}$ is the cardinal of retained prototypes of class c. In order to speed up the execution time we used a CUDA GPU implementation of Nearest Neighbor search [14].

Classification results are summarized in Fig. 3, where the mean Average Precision (MAP %) over all test images is shown for different prototype sets. We computed MAP as the average of diagonal entries in the confusion table, whereas the size of prototype set is reported as θ, that is the ratio of the number of retained prototypes and the overall size of training data. Fig. 3 also reports results of vanilla k-NN (with random sampling of the prototypes from the training data). We observe that the improvement over regular k-NN is dramatic, even when decreasing the prototype number. E.g., only 14% of prototypes allow for 20% improvement over classic k-NN. (See the marked points in the figure.) Besides this precision improvement, MLNN also enables to drastically reduce the computational complexity with respect to plain k-NN (gain up to a factor 4 when discarding half prototypes).

Finally, the confusion table reported in Fig. 4 highlights the difficulty of discriminating between couples of visually similar object categories, like "cup" and "ewer". Moreover, most of mistakes may be due to an insufficient representation of an object category in the prototype set. Namely, categories with few prototype descriptors, like "motorbikes", are more likely to be confused with over-represented categories (e.g., "accordion"). Normalizing the number of prototypes per class, e.g., by adapting the resolution of dense descriptors to the actual object size, is expected to improve classification rate in such categories [3].

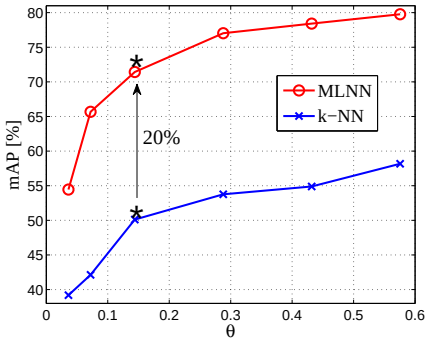

Fig. 3. MLNN classification performances in terms of MAP as a function of the proportion θ of retained prototypes.

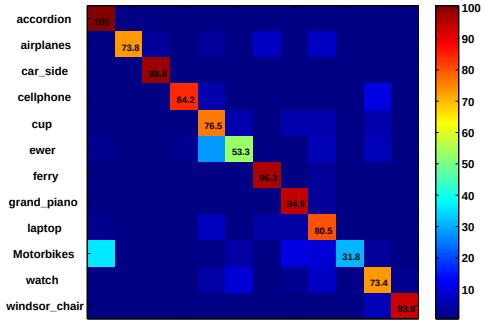

Fig. 4. Confusion table when retaining all prototypes with positive leveraging coefficients ($\theta = 0.58$, $k = 11$).

4. REFERENCES

[1] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[2] J. Sivic and A. Zisserman, "Video Google: Efficient visual search of videos," vol. 4170 of *LNCS*, pp. 127–144. 2006.

[3] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial Pyramid Matching for recognizing natural scene categories," in *CVPR*, 2006, pp. 2169–2178.

[4] H. Zhang, A. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *CVPR'06*, 2006, pp. 2126–2136.

[5] Trevor Hastie and Robert Tibshirani, "Discriminant adaptive nearest neighbor classification," *PAMI*, vol. 18, no. 6, 1996.

[6] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, "Information-theoretic metric learning," in *ICML*, 2007, pp. 209–216.

[7] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining and Knowledge Disc.*, vol. 6, pp. 153–172, 2002.

[8] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, pp. 297–336, 1999.

[9] H. Zou, J. Zhu, and T. Hastie, "New multicategory boosting algorithms based on multicategory fisher-consistent losses," *Annals of Applied Statistics*, vol. 2(4), pp. 1290–1306, 2008.

[10] R. Nock and F. Nielsen, "Bregman divergences and surrogates for learning," *PAMI*, vol. 31, pp. 2048–2059, 2009.

[11] Y. Freund and R. E. Schapire, "A Decision-Theoretic generalization of on-line learning and an application to Boosting," *Journal of Computer and System Sciences*, vol. 55, 1997.

[12] P. Piro, R. Nock, F. Nielsen, and M. Barlaud, "Boosting k-nn for categorization of natural scenes," *ArXiv:1001.1221*, 2009.

[13] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *ECCV*, 2008, vol. I, pp. 304–317.

[14] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using gpu," in *CVPR Workshop on Computer Vision on GPU (CVGPU)*, Anchorage, Alaska, USA, June 2008.